

---

EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE  
POLITECNICO FEDERALE DI LOSANNA  
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

---

**COMMUNICATION SYSTEMS DIVISION (SSC)**  
CH-1015 LAUSANNE, SWITZERLAND  
<http://sscwww.epfl.ch>



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE



## **A Survey of Distributed Network and Systems Management Paradigms**

Jean-Philippe Martin-Flatin, Simon Znaty and Jean-Pierre Hubaux

Version 1: December 1997

Version 2: August 1998

Technical Report SSC/1998/024

# A Survey of Distributed Network and Systems Management Paradigms

Jean-Philippe Martin-Flatin<sup>1,2</sup>, Simon Znaty<sup>2,3</sup> and Jean-Pierre Hubaux<sup>2</sup>

---

**ABSTRACT:** Since the mid 1990s, network and systems management has steadily evolved from a centralized paradigm, where all the management processing takes place in a single management station, to distributed paradigms, where management is distributed over a potentially large number of nodes. Some of these paradigms, epitomized by the SNMPv2 and CMIP protocols, have been around for several years, whereas a flurry of new ones, based on mobile code, distributed objects or intelligent agents, only recently emerged. The goal of this survey is to classify all major network and systems management paradigms known to date, in order to help network and systems administrators design a management application. In the first part of the survey, we present a *simple typology*, based on a single criterion: the organizational model. In this typology, all paradigms are grouped into four types: centralized paradigms, weakly distributed hierarchical paradigms, strongly distributed hierarchical paradigms and cooperative paradigms. In the second part of the survey, we gradually build an enhanced typology, based on four criteria: delegation granularity, semantic richness of the information model, degree of specification of a task, and degree of automation of management. Finally, we show how to use our typologies to select a management paradigm in a given context.

---

**KEYWORDS:** Distributed Network Management, Distributed Systems Management, Integrated Management, Mobile Code, Distributed Objects, Intelligent Agents, Typology.

---

## 1. INTRODUCTION

Network and systems management has thrived on either centralized or weakly distributed paradigms for many years. Soon after the advent of open systems in the second half of the 1980s, proprietary solutions gradually gave way to two open protocols, the Simple Network Management Protocol (SNMP) [9] and the Common Management Information Protocol (CMIP) [15], in the early 1990s. These protocols primarily addressed what was then perceived as the most critical feature lacking in existing network and systems management systems: interoperability between multiple vendors. SNMP was widely adopted by the Internet Protocol (IP) world to manage local-area networks, wide-area networks and intranets, and, to a lesser extent, to manage distributed systems. In parallel to this wide-scale deployment, CMIP, richer but more complex than SNMP, found a niche market in the telecommunications world, as the International Telecommunication Union, Telecommunication Standardization Sector (ITU-T) decided to adopt the Open Systems Interconnection (OSI) management model [12, 13, 75] as the basis for its Telecommunications Management Network (TMN) model [32, 56].

Despite the lack of competition between these two protocols, which looked set to rule their separate markets for many years, the use of both SNMP and CMIP has been questioned since the mid 1990s, together with their underlying models. Why is it that more and more network managers are now demanding strongly distributed management technologies, when the same people were happy with centralized or weakly distributed technologies a few years ago?

The answer, in our view, is twofold. First, strongly distributed management paradigms address some of the major shortcomings of traditional paradigms: beyond mere interoperability, they offer scalability, flexibility and robustness [26]. These three features, identified by Goldszmidt to motivate the use of his own model, Management by Delegation (MbD), can actually justify the use of any kind of strongly distributed management technology. Second, much progress was made in software engineering since CMIP and SNMP were devised, and new technologies suggested new ways of doing network and systems management. For example, the architecture of distributed objects offered by the Common Object Request Broker Architecture (CORBA) [52] of the Object Management Group (OMG) proved to be a viable alternative to the traditional client-server paradigm, while intelligent agents [74] started to spread from Distributed Artificial Intelligence (DAI) to distributed systems. New languages also appeared, such as Java [27], widely adopted by the Web community, or the Knowledge Query and Manipulation Language (KQML) [21], well established in Multi-Agent Systems (MAS), a sub-domain of DAI.

---

1. Correspondence should be directed to J.P. Martin-Flatin, EPFL-DI-ICA, 1015 Lausanne, Switzerland. Email: martin-flatin@epfl.ch. Phone: +41-21-693-4668. Fax: +41-21-693-6610.

2. Institute for computer Communications and Applications (ICA), Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland.

3. Multimedia Networks and Services Dept, École Nationale Supérieure des Télécommunications de Bretagne (ENST-Bretagne), 35512 Cesson Sévigné, France.

As a result, the network and systems management community was recently overwhelmed by an avalanche of new technologies. Today, research is going in all directions, and it is increasingly difficult to tell in which one network and systems management is currently heading. Designers of network and systems management applications are faced with increasingly difficult decisions to make: what management paradigm should I use to manage my network or my distributed system? Should I choose a new management technology, thereby gaining in functionality, but losing out in terms of standardization, and exposing my production network to dreaded heterogeneity problems? Supposing I have decided to adopt a new management paradigm, what technology should I go for, when dozens of them have been prototyped, but hardly any has ever been deployed in real life to manage production networks or systems? Should I use a mix of different paradigms or a single one? What about a single paradigm but multiple technologies?

The goal of this article is to clarify the situation for designers of network and systems management applications, by classifying all technologies into a limited set of paradigms, and by proposing criteria to assess and weight the relative merits of different paradigms or technologies. In particular, we show that there is no win-all solution: different technologies are good at managing different networks and distributed systems.

In this survey, we propose two ways of categorizing network and systems management paradigms; we call them the *simple typology* and the *enhanced typology*. Prior to presenting these typologies, we first define a terminology in section 2, since there is a great deal of inconsistency in the terminology used by different authors in the network and systems management community, in the software engineering community, or in the DAI community. In section 3, we present the simple typology, based on a single criterion: the organizational model. In this typology, all paradigms are grouped into 4 broad types: centralized paradigms, weakly distributed hierarchical paradigms, strongly distributed hierarchical paradigms and cooperative paradigms.

The remainder of this article is dedicated to the enhanced typology. In section 4, we draw a parallel between organization structures found in the enterprise world and those considered in network and systems management; we delineate a common trend between these two worlds, and show that the delegation granularity is a critical criterion for our typology. We then introduce the concepts of micro-task and macro-task. In section 5, we study the three other criteria retained for our enhanced typology: the semantic richness of the information model, the degree of automation of management and the degree of specification of a task. This leads us to our enhanced typology, presented in section 6. In section 7, we give examples of how to use this typology to select a management paradigm for a given network or a given distributed system. Finally, we present some related work in section 8, and our own future work in section 9.

## 2. TERMINOLOGY

Before we proceed with the review of distributed network and systems management paradigms, we must first acknowledge that the distributed network and systems management research community has not fully converged on a common terminology yet. Most people agree that the centralized paradigm is characterized by a single Network Management Station (NMS), concentrating all the management application processing, and a collection of agents limited to the role of dumb data collectors; but there are different views on several other definitions. For example, some authors advocate the use of their new distributed paradigm by criticizing the centralized paradigm, but overlook hierarchical paradigms [26, 37]; and most of them simply ignore the cooperative paradigm [25, 30, 35, 37, 46, 60, 63]. To address this confusion, we therefore propose the following terminology.

Decentralized management is to the enterprise world what distributed management is to computer science: a management paradigm based on the delegation of tasks to other entities. These entities are persons in the enterprise world, machines or programs in computer science. *Delegation* is a generic word, used in both contexts to embody the process of transferring power, authority, accountability and responsibility [19, 49] for a specific task to another entity. In distributed network and systems management, delegation always goes down the management hierarchy: a manager at level (N) delegates a task (i.e., a management processing unit) to a subordinate at level (N+1); this is known as *downward delegation*. In the enterprise world, we can also find *upward delegation*; for example, an employee delegates his tasks to his manager when he is off sick [49]. Downward delegation and upward delegation are two kinds of *vertical delegation*, typical of hierarchical paradigms. A *hierarchical paradigm* is characterized by a multi-layer pyramid, comprising a *top-level manager* (at level 1), several *mid-level managers* (at levels 2, 3...), and *operatives* at the lowest level [19]. In network and systems management, NMSs globally refer to the top-level and mid-level managers, whereas operatives are called *agents*. Orthogonally to vertical delegation, we have *horizontal delegation*, between two peers at the same level, typical of *cooperative paradigms* used in Distributed Artificial Intelligence (DAI). Distributed network and systems management relies on either

an underlying hierarchical paradigm, a cooperative paradigm, or a combination of the two: indeed, any paradigm outside the realm of centralized paradigms belongs to distributed network and systems management.

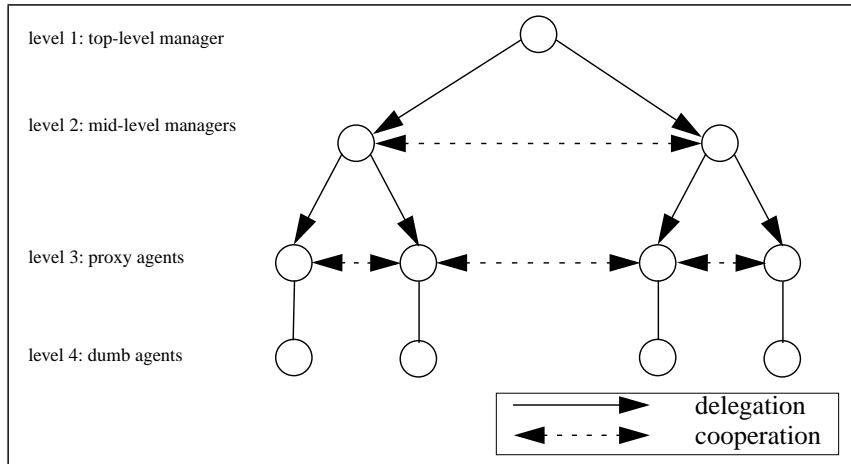


Fig. 1. Delegation and cooperation

Delegation is normally a *one-to-one relationship*, between a manager and an agent in a hierarchical management paradigm, or between two peers in a cooperative management paradigm. Arguably, delegation may also be considered, in some cases, as a *one-to-many relationship*, where a task is delegated to a group of entities, collectively responsible for the completion of the task. One-to-many delegation is forbidden by most authors in enterprise management [49, 71, 2, 19]. It could be envisaged in DAI though. In distributed network and systems management, we propose to classify it as a form of cooperation, by coupling hierarchical and cooperative paradigms: a manager delegates a task to an agent, and this agent in turn cooperates with a group of agents to achieve this task. In the case of a *many-to-many relationship*, we are clearly in the realm of cooperation rather than delegation.

The meaning of NMS has shifted over the years from *Network Management System* to *Network Management Station*. The reason for this is clear: SNMP, when it was first released, assumed an underlying centralized paradigm, characterized by a single network management station. The whole network management system was made of a management application running on a single workstation. Several years later, SNMPv2 adopted a hierarchical paradigm, *a la* CMIP, where the network management system is distributed over multiple stations. Since we are now clearly in the days of distributed management, we will translate NMS into *Network Management Station* throughout this article.

To cope with legacy systems, whose internal SNMP/CMIP agent does not support the capabilities described in strongly distributed paradigms, we assume in this article that such systems make use of proxy agents if necessary. A *proxy agent* is a management gateway, dedicated to a certain legacy system and external to it; it is located between the manager and the SNMP/CMIP agent, and is transparent to the management application. It can for instance translate a CORBA request into SNMP/CMIP protocol primitives, and vice versa. When a proxy agent is used, the SNMP/CMIP agent embedded in the legacy system is called a *dumb agent*. Throughout this article, when we refer to an *agent*, we may as well refer to the pair {dumb agent, proxy agent} or simply to the SNMP/CMIP agent.

This proxy agent is sometimes called a *delegated agent*. This expression is ambiguous, since some authors give this name to programs remotely transferred to an agent [26], so we will avoid using it. To conclude with agents, the word *agent* has traditionally had a different meaning in the DAI and distributed network and systems management communities. In order to avoid any confusion, we will speak of an *intelligent agent* when we mean an agent in the DAI sense in this article.

Some people confuse *management paradigms* and *management technologies*. In the tradition of software engineering, and specially object-oriented analysis and design, we consider that technologies implement paradigms [23]. At the analysis phase, network and systems administrators select a management paradigm. At the design phase, they select a management technology. At the implementation phase, they use that technology.

The meaning of *enterprise management* is also ambiguous. To most people, this is something you learn doing business studies, and this is indeed the sense we have retained in this article. But a new fashion recently appeared in computer science, epitomized by Web-Based Enterprise Management (WBEM). Particularly in large, geographically dispersed corporations, the problem is not so much managing networks and systems *per se*: it is mostly to manage networks and

systems and services that are deeply intertwined. Before the days of WBEM, this was commonly referred to as *integrated management*. Since then, the main marketing target has shifted from networks and systems to services, specially multimedia services, so the marketing terminology changed. But in our view, there is nothing more to enterprise management than there is already in integrated management. Since the latter is unambiguous, and since we see no reason beyond marketing to change a well-accepted terminology, we will not use enterprise management in its new, ambiguous sense.

Finally, to conclude with the terminology, Meyer [46] already attempted to review some management paradigms, but used the word *taxonomy* rather than *typology*. According to the New Encyclopaedia Britannica, a taxonomy is “*in a broad sense, the science of classification, but more strictly the classification of living and extinct organisms, i.e., biological classification*” [50]; whereas a typology is a “*system of groupings (such as ‘landed gentry’ or ‘rain forests’), usually called types, the members of which are identified by postulating specified attributes that are mutually exclusive and collectively exhaustive[...]. A type may represent one kind of attribute or several, and need include only those features that are significant for the problem at hand*” [51]. Conversely, classifications “*deal with ‘natural classes’, i.e., with groupings that differ from other groupings in as many particulars as one can discover*” [51]. Based on these definitions, the work that we are striving to achieve in this article is clearly a typology, not a taxonomy. One could argue that only the literal sense of the word *taxonomy* is given above, whereas many people use it in a figurative sense nowadays; in that case, the words *typology* and *taxonomy* could be considered as synonymous.

### 3. A SIMPLE TYPOLOGY OF NETWORK AND SYSTEMS MANAGEMENT PARADIGMS

With these definitions in mind, we are now going to present our simple typology of network and systems management paradigms. When we built it, we tried to meet six objectives:

- it should be a first, intuitive categorization of management paradigms
- for the sake of clarity, it should comprise a limited number of types
- it should clearly separate centralized paradigms from distributed paradigms
- it should highlight what is inherently different between traditional paradigms and new paradigms
- it should distinguish paradigms relying on vertical delegation from those based on horizontal delegation
- finally, based on it, designers of network and systems management applications should find at a glance what paradigm is implemented by a given technology.

To keep this typology simple, we decided to base it on a single criterion, the organizational model, like most authors do (see section 8). To meet the third objective, we started with two types: centralized paradigms and distributed paradigms. To meet the fourth objective, we had to further split up distributed paradigms. By studying the different technologies implementing distributed management, we found that regardless of their idiosyncrasies, all of these technologies could be classified in two broad types, according to the role played by agents in the management application. We called them weakly and strongly distributed technologies; they implement respectively weakly and strongly distributed paradigms.

*Weakly distributed paradigms* are characterized by the fact that network management processing is concentrated in a handful of NMSs, whereas the numerous agents are limited to the role of dumb data collectors (in an intranet, we typically have one or two orders of magnitude between the number of agents and the number of NMSs). Typical examples of weakly distributed network management are the hierarchical models incarnated by CMIP and SNMPv2. *Strongly distributed paradigms*, on the other hand, decentralize management processing down to each and every agent: management tasks are no longer confined to NMSs, all agents and NMSs take part in the network management application processing. Many strongly distributed technologies have been suggested in the recent past, which can be grouped into four types. The first three, mobile code, distributed objects and WBEM, are based on vertical delegation, and thus assume an underlying hierarchical paradigm. The fourth type, intelligent agents, is based on horizontal delegation, and assumes a cooperative paradigm.

The simple typology that we propose is comprised of four types:

- centralized paradigms
- weakly distributed hierarchical paradigms
- strongly distributed hierarchical paradigms
- cooperative paradigms.

In this form, it meets our first five objectives. We are now going to present each paradigm in more detail, and review the main technologies implementing them. Once all paradigms and technologies have been presented, we will summarize our simple typology in a synthetic diagram, page 11: this will allow us to meet our sixth and last objective.

### 3.1. Traditional Paradigms

The first two types of our typology, *centralized paradigms* and *weakly distributed hierarchical paradigms*, constitute what is usually referred to as *traditional paradigms*. These paradigms are those currently used to manage real-life networks and systems. They rely on the CMIP protocol or on one of the SNMPv\* protocols. Both centralized paradigms and weakly distributed hierarchical paradigms share a common property: the semantics offered to the designer of a network and systems management application are entirely dependent upon the communication protocol used underneath. In other words, the abstraction levels used to build a management application have a one-to-one mapping to the protocol primitives. This is not inherent in the protocols themselves, but due to the way they are traditionally used. We will come back to this point in section 5.1.

#### 3.1.1. Centralized Paradigms

The first release of SNMP [9], now commonly referred to as SNMPv1, is the typical technology implementing a centralized paradigm. This protocol is well-known, and its management framework is presented by many authors [55, 63, 60, 30, 35]. With the whole family of SNMPv\* protocols, the Internet Engineering Task Force (IETF) uses the same name for the underlying protocol and the management framework, unlike the ISO with CMIP and OSI, for instance. This confusion has been acknowledged, and recent documents [29] now refer to the *SNMPv1 framework* or the *SNMPv1 protocol* explicitly.

As we will see in next section, the IETF later released a new version of this protocol, SNMPv2, supporting a weakly distributed hierarchical paradigm. Its acceptance was fairly poor, largely because its security framework and the Manager-to-Manager Management Information Base (MIB) [10] proved to be unworkable in deployment [43]. The SNMPv2 working group at IETF could not agree on new administrative and security frameworks, and its work appeared to be stalled for a while; by and large, people kept using SNMPv1. But besides these controversial frameworks, SNMPv2 had done some good, by codifying a number of existing practices which were left unspecified by SNMPv1. To take advantage of this work, it was decided, as a last minute compromise [43], to issue a downgrade of SNMPv2: SNMPv2c [11], also known as the *community-based SNMPv2*. The concept of *party* was left out, which rendered the Manager-to-Manager MIB obsolete, and made hierarchical management with multiple levels of managers impossible. SNMPv2c is therefore classified as a technology implementing a centralized paradigm.

Most intranets are managed with SNMPv1 today, whereas wide-area networks are generally managed with either SNMPv1 or SNMPv2c. Most of the sites which migrated from SNMPv1 to SNMPv2 now use SNMPv2c instead.

#### 3.1.2. Weakly Distributed Hierarchical Paradigms

Three independent evolutions soon exposed a major weakness in the centralized paradigm, and with it SNMP: scalability. First, throughout the 1990s, the IP world has been expanding at a very fast pace. Once limited to Unix machines, the IP protocol stack (which includes the Transmission Control Protocol, TCP, the User Datagram Protocol, UDP, and SNMP) became available on most network devices in the early 1990s, and on most PCs and Macintoshes in the mid 1990s, thanks largely to the success of the Web. Today, it is virtually ubiquitous. Second, the size of networks grew dramatically, as the number of installed machines was growing fast, and the proportion of networked machines (as opposed to stand-alone) was also increasing quickly. Third, the size of SNMP MIBs increased several times: IP routers and hubs just had a few FDDI, Ethernet and Token Ring ports to manage a few years ago; but now, Asynchronous Transfer Mode (ATM) switches and intelligent hubs have many more entities to manage (ports, cross connections...), and require much more data to be brought back to the NMS.

So the very success of SNMPv1 was the cause of its decline: it was good at managing relatively small networks, but could not scale to large networks (e.g. geographically dispersed enterprises), and could not cope with ever more management data. A new paradigm was needed to tackle scalability. People turned to weakly distributed hierarchical paradigms, as CMIP, a protocol used primarily by network operators in the telecommunications world, had already shown how to solve this problem: by using a hierarchical organizational model, with multiple levels of managers.

### 3.1.2.1. RMON, SNMPv2 and SNMPv3 Management Frameworks

Remote MONitoring (RMON) probes were the first and simplest form of delegation added to SNMPv1. The RMON MIB was issued in 1991 [68], and updated in 1995 [69] as RMON was getting more and more integrated into intelligent hubs. By gathering usage statistics in these probes, network administrators could delegate simple network management tasks, thereby relieving the NMS from the corresponding processing burden, and decreasing the amount of management data to move about. Tasks achieved by RMON are static, in the sense that only the gauges and traps hard-wired in the RMON MIB are available, together with all kinds of combinations thereof (via the *filter* mechanism). If contact is lost between the RMON-capable agent and the central NMS, statistics are still gathered, but no independent corrective action may be undertaken by the agent. RMON can be considered as a proxy agent mechanism, operating on behalf of multiple agents. It is primarily aimed at managing large intranets.

Support for a fully-fledged hierarchical paradigm was added to SNMP in a new version of the protocol, SNMPv2, first released in 1993 [24] and reissued in 1996 [44]. This protocol is also well-known, and presented by many authors [55, 63, 60, 30, 35]. Among the three MIBs defined as part of the SNMPv2 specification, one is in charge of distributed management: the Manager-to-Manager MIB [10]. If contact is lost between a mid-level manager and the top-level manager, independent corrective action can be undertaken by the mid-level manager. But if contact is lost between a mid-level manager and an agent, the agent is left on its own. SNMPv2 is primarily targeted at geographically dispersed enterprises, but is hardly used in practice.

The IETF is currently defining a new architecture for SNMP frameworks [29], which should support cohabitation and interoperability between existing SNMP frameworks (SNMPv1, SNMPv2 and SNMPv2c), and better handle the evolution of SNMP (SNMPv3 and its descendants). The SNMPv3 framework, which is still in the making, will supplement the SNMPv2 framework by supporting a new message format, better security for messages, and access control [29]. This new architecture is also expected to better specify engineering details of dual-role entities, also known as SNMP mid-level managers, and make it possible to distribute management in the Internet in practice. Whether hierarchical management will become a reality with SNMPv3 remains to be seen.

### 3.1.2.2. OSI Management Framework and TMN

Unlike SNMP, which was deployed in many sectors of activity, CMIP found a niche market in the telecommunications world, where it is used to manage both networks and systems. The OSI management framework [12, 13, 75], CMIP [15], the Common Management Information Service (CMIS) [14] and the Guidelines for the Definition of Managed Objects [16] are well-known and presented by many authors [63, 30, 35]. In 1992, the ITU-T adopted the OSI management model as the basis for its TMN model [32, 56], which mandates the use of CMIP and CMIS. TMN is therefore based on a weakly distributed management paradigm.

One of the management services offered by CMIS is M\_ACTION. This service is hardly ever used to its full potential in practice, but is conceptually richer than RMON: any agent can execute a static, pre-defined task when requested by the NMS. We get here a flavor of strongly distributed management.

## 3.2. Strongly Distributed Hierarchical Paradigms

Weakly distributed hierarchical paradigms address the main shortcoming of centralized models, scalability, but they also show a number of limitations in practice. First, they lack robustness: if contact is lost between the agent and the manager (e.g., due to a network link going down), the agent has no means to take corrective action in case of emergency. Second, they lack flexibility: once a task has been defined in an agent (via RMON, or CMIP/CMIS with M\_ACTION), there is no way to modify it on the fly; it remains static. Third, they are expensive: most of the management application processing still takes place in NMSs, which are considerably more expensive than agents.

To address this, a new breed of technologies emerged, based on strongly distributed hierarchical paradigms. The full potential of large-scale distribution over all managers and agents was first demonstrated in network and systems management by Goldszmidt with his Management by Delegation (MbD) framework [25, 26], which set a milestone in this research field. The novelty of this work stems on the simple, yet insightful idea that with the constant increase in processing power of every computer system and network device, network and systems management no longer ought to be limited to a small set of powerful management stations: all agents could get involved, and become active in the management application. For the first time with MbD, network devices were suddenly promoted from dumb data collectors to the rank of managing entities.

MbD triggered a lot of research work in strongly distributed network and systems management. The impact of the novel concepts it brought to this research community was leveraged by the emergence of many promising technologies at about the same time in other research communities, some coming from software engineering, others from distributed applications, others from the object-oriented community. We are now going to present the paradigms underlying these strongly distributed technologies in more detail. These paradigms are grouped in two broad types: mobile code and distributed objects.

### 3.2.1. Mobile Code

Mobile code paradigms encompass a vast collection of very different technologies, all sharing a single idea: to provide flexibility, one can dynamically transfer programs into agents, and have these programs executed by the agent. The program transfer like the program execution can be triggered by the agent itself, or by an entity external to the agent such as a manager or another agent.

Fuggetta et al. [23] made a detailed review of mobile code, where they clearly define the boundaries between technologies, paradigms (what they call *design paradigms*) and applications. As far as mobile code technologies are concerned, they define *strong mobility* as the ability of a Mobile Code System (MCS) to allow an execution unit (e.g., a Unix process or a thread) to move both its code and its execution state to a different host: the execution is suspended, transferred to the destination host, and resumed there. *Weak mobility*, on the other hand, is the ability of an MCS to allow an execution unit on a host to bind dynamically code coming from another host: the code is mobile, but the execution state is not preserved automatically by the MCS (though it is still possible to program this preservation explicitly, of course).

By analyzing all existing MCSs, the authors identified three different types of mobile code paradigms:

- *Remote Evaluation* (REV): when a client invokes a service on a server, it does not only send the name of the service and the input parameters: it also sends the code along. So the client owns the code needed to perform the service, while the server owns the resources and provides an environment to execute the code sent by the client. REV can be regarded as an extension of Remote Procedure Calls (RPCs).
- *Code On Demand* (COD): a client, when it has to perform a given task, contacts a code server, downloads the code needed from that server, links it in on the fly (dynamic code binding) and executes it. So the client owns the resources and the server owns the code.
- *Mobile Agent*<sup>1</sup> (MA): an MA is an execution unit able to migrate autonomously to another host and resume execution seamlessly. Conceptually, an MA can migrate its whole virtual machine from host to host: it owns the code, not the resources.

A number of technologies can be used to implement these three paradigms. Some of them are just languages, others complete systems potentially including a virtual machine, a secure execution environment, etc. Telescript, Tycoon, Agent Tcl and Emerald are examples of strong MCSs, whereas Java, Mole, Tacoma, M0, Facile, Obliq and Safe-Tcl are examples of weak MCSs (see references in [23]). Indeed, some technologies can be used to implement several paradigms [23].

Mobile code paradigms were first used in network and systems management by Goldszmidt and Yemini, when they devised the Manager-Agent Delegation (MAD) model in 1991 [76]. This management framework was later enhanced and renamed Management by Delegation (MbD); it was fully specified in 1995 [25]. MbD is well-known and summarized in [26]. It is a mixture of the REV paradigm (to send delegated agents to elastic servers) and the client-server paradigm (to remotely control the scheduling and execution of delegated agents).

Since 1995, MCSs have encountered a growing success in network and systems management, with people like Magedanz et al. [37] or Baldi et al. [3]. In 1996, two working groups were created, one by IETF and another by ISO [57], in order to integrate mobile code concepts in their respective management frameworks. So far, this has resulted in an Internet draft defining the Script MIB [36], and an ITU-T draft (X.753, announced but not available yet) defining the Command Sequencer management function. In 1997, by studying theoretically the network traffic generated by MCSs implementing

---

1. The choice of the phrase *mobile agent* may appear a bit unfortunate, but alas reflects a clash in the terminologies used by the distributed applications and DAI communities. This clash has confused a number of people [34, 53, 47, 64], who liken (at different degrees) the concepts of mobile code, mobile agent and intelligent agent. In DAI, a mobile agent is a fully-blown intelligent agent, as we define it in section 3.4, with an extra property: mobility. In this sense, there is much more to a mobile agent than just a mobile program and a mobile state.



REV, COD and MA paradigms, Baldi and Picco [4] made a quantitative evaluation of the effectiveness and suitability of mobile code paradigms in network management.

One area where mobile code have recently been put to a test is known as *active networks*. This new concept was proposed in 1995 by Tennenhouse and Wetherall [67], and first adapted to network management by Yemini in 1996 [77]. By definition, “active networks allow their users to inject customized programs into the nodes of the network” [67]. These programs may perform customized computations on the user data flowing through them, and possibly alter this data (e.g., compress it). This breaks the principle that transport networks should carry user data opaquely.

There are two approaches to active networks. The evolutionary path, called the *programmable switch approach*, keeps the existing packet format and provides a mechanism for downloading programs to dynamically programmable nodes [61, 77]. The revolutionary path, also known as the *capsule approach*, considers packets as miniature programs that are encapsulated in transmission frames, and executed at each node along their path [67]. Flexible and robust management is very natural with active networks, specially in the areas of network monitoring and event filtering [66, 77], as monitoring programs are dispatched through the network. These programs are high-level filters that watch and instrument packet streams in real time. They maintain counters, and report results back to the NMS.

### 3.2.2. Distributed Objects

Independently of mobile code, a second type of strongly distributed hierarchical paradigms has recently emerged, based on distributed object technologies. We will present the four main ones in this section: Sun’s Java Management Application Programming Interface (JMAPI), CORBA, WBEM and the Open Distributed Management Architecture (ODMA). Unlike mobile code technologies, which come from both academia and industry, distributed object technologies all come from industry.

#### 3.2.2.1. JMAPI

Since most enterprises buy vendor-specific, NMS-dependent add-ons like CiscoWorks to manage their network equipment, some people suggested to save the cost of the NMS (where both the software and the underlying Unix workstation or Windows NT server are expensive) by using Web browsers on cheap PCs. To do so, equipment vendors, rather than support multiple platforms for each add-on, need only provide a single device-specific, platform-independent management applet, the so-called *embedded management application* [72]. The applet, written in Java, offers a GUI very similar to the add-ons, and allows a network administrator to manage a network device with Java Remote Method Invocation (RMI), rather than SNMP. Sun made publicly available a set of tools and guidelines to build these applets: JMAPI [65]. In this architecture, vendors save the cost of supporting many add-ons on multiple platforms, and the loss of revenue incurred by scrapping add-ons is covered by selling embedded HyperText Transfer Protocol (HTTP) servers on a per-device basis. This solution, based on Java and the Web, is promoted by Sun, IBM and many network equipment vendors.

#### 3.2.2.2. CORBA

The OMG, faced with the issue of interoperability in the object-oriented world, addressed it by standardizing the Object Management Architecture, often referred to by its main component: CORBA [52, 59]. Since OSI is object-oriented and SNMP entities map easily onto objects, it took little time for researchers to start integrating CORBA with existing network management environments.

The Joint Inter-Domain Management (JIDM) group, jointly sponsored by the X/Open and the Network Management Forum (NM Forum), was created to provide tools that enable management systems based on CMIP, SNMP and CORBA to work together. The SNMP/CMIP interoperability had previously been addressed by the ISO-Internet Management Coexistence (IIMC) group of the NM Forum, which specified the translation between the SNMP and CMIP services, protocols and information. Both CMIP/CORBA and SNMP/CORBA [40] interworking were tackled by JIDM, which addressed specification translation and interaction translation. Algorithms were defined for the mapping between GDMO/ASN.1 and CORBA IDL [41], and between SNMP MIBs and CORBA IDL [42]. The JIDM mappings allow CORBA programmers to write OSI or SNMP managers and agents without any knowledge of GDMO, ASN.1 and CMIP, and conversely GDMO, CMIS or SNMP programmers to access IDL-based resources, services or applications without knowing IDL.

CORBA 2.0 [52] was released in 1995. It received a wide acceptance in the distributed applications community, but also in telecommunications, where it is gradually becoming a *de facto* standard, a rarity in this industry traditionally based on *de jure* standards. The Telecommunications Information Networking Architecture Consortium (TINA-C) [5] selected CORBA for its distributed processing environment in 1996, and most telecommunication equipment vendors are gradually incorporating CORBA to manage their switches.

### 3.2.2.3. WBEM

Open standards like SNMP and CMIP have virtually killed off proprietary management solutions in the fields where open systems encountered a large success, namely the Internet, intranets and telecommunications. But in the rest of the industry, where large PC networks are predominant, proprietary management platforms are still the rule and open platforms the exception. Today, most desktops are managed with proprietary protocols like Novell Netware or Microsoft Windows NT.

A few years ago, the Desktop Management Task Force (DMTF) issued the Desktop Management Interface (DMI) specification [17], and promoted open management for desktops. Unfortunately, it has encountered little success so far. Things could change though; the same companies which did not bother too much about open management a few years ago, and invested in separate platforms to manage their network equipment and PC systems, are now facing the costs incurred by this lack of interoperability, and pushing the industry to integrate network and systems management within a single platform.

To address this need, WBEM [7], a management framework promoted by a consortium led by Microsoft, takes a radical approach. It proposes a new object model, the HyperMedia Management Schema (HMMS), a new protocol, the HyperMedia Management Protocol (HMMP), and a new environment to manage elements as objects, the HyperMedia Object Manager (HMOM). The DMTF is currently specifying schemata for the Common Information Model (CIM) [18], based on HMMS. It is also working on SNMP/CIM, DMI/CIM and CMIP/CIM proxies, in order to integrate WBEM with existing protocols and object models.

The main challenges here are interoperability and integration. Whether the industry will accept to go for new protocols and new information models, a few years after huge investments were put in open solutions like SNMP or CMIP, or in proprietary solutions like Novell Netware, remains to be seen. It seems unlikely, for instance, that this will happen in the near future in the telecommunications industry. The fact that this new management framework is backed by Microsoft, which has traditionally been defending its own proprietary solutions, and now claims to promote open management, is also a cause of concern. But many vendors seem to believe in this revolutionary path, and the industrial consortium now backing Microsoft is very large.

### 3.2.2.4. ODMA

The purpose of ODMA [31] is to extend the OSI management architecture, and thus the TMN architecture, with the Reference Model of the ISO Open Distributed Processing (RM-ODP) framework, which provides for the specification of large-scale, heterogeneous distributed systems. This joint effort of the ISO and the ITU-T has led to a specialized reference model for the management of distributed resources, systems and applications. It is based on an object-oriented distributed management architecture, composed of computational objects. These objects offer several interfaces, some for the purpose of management, others for different purposes.

In ODMA, there are no longer managers and agents with fixed roles, like in the OSI management framework. Instead, computational objects may offer some interfaces to manage other computational objects (manager role), and other interfaces to be managed (agent role). Moreover, by adopting the computational viewpoint of ODP, ODMA also rendered the location of computational objects transparent to the management application (see section 5.1.2): as far as the management application is concerned, computational objects may live anywhere, not necessarily inside a specific agent or NMS. Therefore, agents may execute advanced management tasks, just like NMSs. In short, the ISO and the ITU-T have gone from a weakly distributed management paradigm, with the OSI management framework, to a strongly distributed management paradigm, with ODMA.

## 3.3. Is Web-based Management a Separate Paradigm?

Just before we go on with cooperative paradigms, let us quickly clarify the concept of *Web-based management*, which has been confused recently by much marketing hype. Since the World-Wide Web (WWW) is now ubiquitous, so cheap and so easy to use, many people argued that it should be used in network and systems management. This resulted in very

different approaches, which are collectively grouped under the heading Web-based management. Some of them are based on weakly distributed hierarchical technologies, others on strongly distributed ones. Web-based management therefore overlaps two types in our simple typology, and does not constitute a type *per se*. One thing that all Web-based technologies share though is that they all require an HTTP server be present in every agent; and indeed, many network equipment vendors already offer this feature today.

Web-based weakly distributed hierarchical technologies use HTTP instead of SNMPv\*, or in conjunction with it. The use of HTTP instead of SNMPv\* became realistic with the advent of HTTP 1.1 [20], which supports long-lived TCP connections and is therefore more efficient than UDP-based SNMPv\* for large MIB retrievals [72] (HTTP 1.0 [6] does not support long-lived TCP connections, so it is always less efficient than SNMPv\*). Within HTTP packets, MIB data can be either encoded in a specific MIME type, or embedded in an HTML structured document. Device-specific command lines can also be encoded the same way; so, when commands provided by the command line interface have no SNMPv\* equivalent, there is no longer a need for *expect* scripts to emulate interactive telnet sessions. Network management security, a well-known problem with all SNMPv\* protocols, can also entirely rely on Web security technologies, which a lot of people are currently working on to secure business transactions over the Web. As far as the management application is concerned, the sole difference between traditional and Web-based weakly distributed hierarchical technologies is the transport protocol: nothing changes with respect to the management paradigm.

Two Web-based strongly distributed hierarchical technologies have been in the spotlight for the past year. They are JMAPI and WBEM, which we presented in section 3.2.2.1 and section 3.2.2.3.

### 3.4. Cooperative Paradigms

Unlike centralized and hierarchical paradigms, cooperative paradigms are *goal-oriented*. What does this mean? For example, in REV-based mobile code technologies, agents receive programs from a manager and execute them, without knowing what goal is being pursued by the manager. Managers send agents the ‘how-to’, with a step-by-step *modus operandi* (coded in the program), and keep the ‘why’ for themselves. Agents execute the program without knowing what it is about, they are ‘dumb’. Conversely, with intelligent agents, managers just send the ‘why’, and expect agents to know how to devise the ‘how-to’. In this sense, agents used in cooperative paradigms are ‘intelligent’. Obviously, there is a price to pay for this: cooperative technologies are much more complex to implement than centralized or hierarchical technologies, and consume more resources.

Cooperative paradigms were only recently considered by the distributed network and systems management community. They originate from DAI, and more specifically from Multi-Agent Systems (MASs), where people are modeling complex systems with large groups of intelligent agents. This research field is still fairly recent, so its terminology is still vague. Specifically, there is no consensus on the definition of an intelligent agent. Many authors have strong (and different) opinions about this ([22] listed 11 definitions in 1996!), which does not help. In 1994, Wooldridge and Jennings took a new approach: instead of imposing on others what an intelligent agent should or should not be, they tried to define a core of properties shared by all intelligent agents, and still allowed any other property as application-specific. This approach has encountered a great deal of success, and contributed significantly to the dissemination of MASs outside the realm of DAI. For these authors, intelligent agents (or to be precise, what they call *weak agents*) must exhibit four properties [74]:

- *autonomy*: an intelligent agent operates without the direct intervention of humans, and has some kind of control over its actions and internal state
- *social ability*: intelligent agents cooperate with other intelligent agents (and possibly humans) to achieve their goals, via some kind of agent communication language
- *reactivity*: an intelligent agent perceives its environment and responds in a timely fashion to changes that occur in it
- *pro-activeness*: an intelligent agent is able to take the initiative to achieve its goals, as opposed to solely reacting to external events.

Pro-activeness is a very discriminating property: while most intelligent agent implementations are reactive, only few of them, according to the authors, qualify for pro-activeness, specially outside the AI community. According to us, this is indeed the main difference between mobile agents, coming from the distributed applications community, and intelligent agents, coming from the DAI community.

For Wooldridge and Jennings, optional properties of weak agents include mobility, veracity (intelligent agents do not knowingly communicate false information), and rationality (intelligent agents are not chaotic, they act so as to achieve their goals). Besides that, they also define *strong agents* as weak agents modeled with human-like characters, e.g. by using

Rao and Georgeff's Belief, Desire, Intention (BDI) model [54]. Strong agents are the type of intelligent agents generally used by the DAI community, whereas weak agents are the type generally used by other research communities.

Two years later, Franklin and Graesser [22] compared the approaches taken by many authors, and, like Wooldridge and Jennings, distinguished between mandatory properties and optional properties. For them, intelligent agents must be reactive, autonomous, goal-oriented (pro-active, purposeful), and temporally continuous (an intelligent agent is a continuously running process). Optionally, they can also be communicative (that is, able to communicate, coordinate and cooperate with other agents,), learning (they improve their skills as time goes by, storing information in knowledge bases), mobile, and have a human-like character. In our view, the fact that intelligent agents should be continuously running processes is an important property: it distinguishes intelligent agents from mobile agents (MA-based mobile code technologies).

Since we consider intelligent agents in the context of cooperative paradigms in distributed network and systems management, their ability to communicate, coordinate and cooperate should be a mandatory property, in our view. We therefore propose that in distributed network and systems management, intelligent agents should always be:

- goal-oriented (pro-active)
- autonomous
- reactive
- cooperative (communicative, coordinating)
- temporally continuous

When intelligent agents are cooperative, they are exposed to heterogeneity problems, and therefore critically need standards for agent management, agent communication languages, etc. Two consortia are currently working on such standards: the Foundation for Intelligent Physical Agents (FIPA) and the Agent Society. Among all the agent communication languages that sprang up in DAI [74], one, KQML [21], has encountered a certain success in the distributed network and systems management community.

More and more researchers are now trying to use intelligent agents to manage networks and systems: Post with the manager/agency paradigm [53], Somers with the HYBRID system [62], Zhang who proposes to extend TMN [78], and the flexible agents school initiated by Mountzia [47, 73, 48]. We should remember though that the limits between mobile agents, following a mobile code paradigm, and intelligent agents, following a cooperative paradigm, are sometimes fuzzy.

### 3.5. Synthetic Diagram

Our simple typology is summarized in the following synthetic diagram:

	centralized paradigms	hierarchical paradigms	cooperative paradigms
not distributed	SNMPv1, SNMPv2c		
weakly distributed		RMON, SNMPv2, SNMPv3, OSI management	
strongly distributed		mobile code, distributed objects	intelligent agents

**Fig. 2.** Simple typology of network and systems management paradigms

Throughout section 3, we presented a simple typology dividing up network management paradigms into four broad types, according to a single criterion: the underlying organizational model. We are now going to refine this approach and gradually build an enhanced typology, based on several criteria. The first criterion, delegation granularity, is derived by comparing the organizational models in network and systems management with organization structures considered in enterprise management.

## 4. COMPARISON WITH ENTERPRISE MANAGEMENT

The topology of an enterprise computer network tends to be modeled after its organization chart. The main reason for this is that the persons accountable for the smooth operation of these networks and systems belong to this chart, and it makes

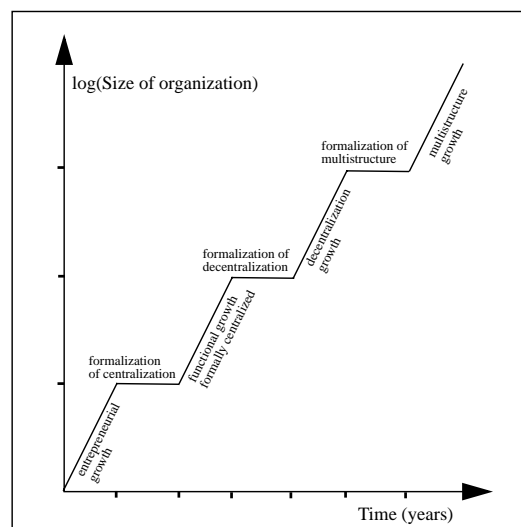
their life a lot easier if different managers have a hold on different computers and network devices. Besides, such a topology is also often justified in terms of budget; sometimes, it also makes technical sense, for instance when different departments are located in different floors or buildings. So network and systems management is not orthogonal to enterprise management: we will show that lessons learned from one research field may benefit the other. Let us now study how delegation works in enterprises, how it maps onto organization structures, and how the two fundamental paradigms (delegation and cooperation) that we identified in network and systems management map into the enterprise world.

#### 4.1. Organization Structures in Enterprise Management

Mullins [49] distinguishes eight ways of dividing work in an enterprise:

- by function (one department per function: production, R&D, marketing, finance, sales... all staff share a common expertise within a department)
- by product (autonomous units, all functions are present in each unit)
- by location (multi-site companies, subsidiaries abroad)
- by nature of the work to be performed (e.g. by security clearance level)
- by common time scales (e.g. shift work vs. office hours work)
- by common processes (e.g. share production facility in manufacturing industry)
- by the staff employed (e.g. surgeons, doctors and nurses in a hospital)
- by type of customer or people to be served (e.g. home vs. export sales).

For Mullins, delegation can take place at two levels: enterprise or individual. At the enterprise level, it is depicted in the organization chart (or at least is supposed to be!), and relies on federal or functional decentralization. He defines *federal decentralization* as “the establishment of autonomous units operating in their own market with self-control and with the main responsibility of contributing profit to the parent body” [49, p. 276]. As for *functional decentralization*, it is “based on individual processes or products” [49, p. 276]. At the individual level, delegation is “the process of entrusting authority and responsibility to others” [49, p. 276] for a specific task.

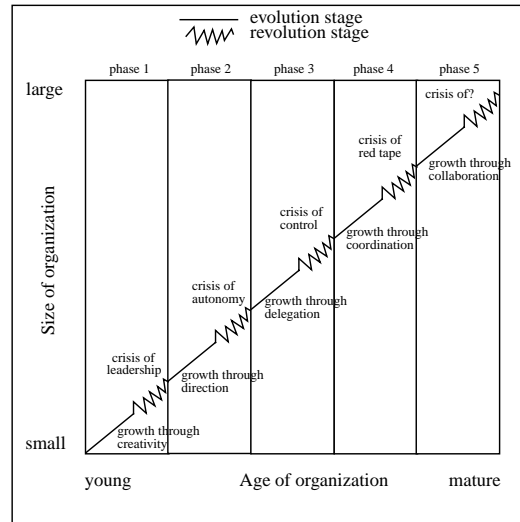


**Fig. 3.** The effect of size on the enterprise structure (adapted from Weinshall [71, pp. 56-57])

Weinshall [71] identifies three basic managerial structures: entrepreneurial, functional and decentralized. The *entrepreneurial structure* is typical of an organization recently created, fairly small and growing fast. It must be managed in an informal and centralized fashion in order to survive. Everything is centered on one person, the entrepreneur who created the company. When organizations grow beyond a certain size, they must go through a major transformation: a whole set of rules by which the work is managed and carried out need be formalized, “in order to cope with the growing quantities of product and services, their variety, and the complexity of the organization” [71, p. 55]. This is called the *functional structure*. The chief executive directly controls the various functional heads, such as the production manager, the marketing manager, the sales manager... The formalized and centralized nature of the functional structure must, at some point, give place to the *decentralized structure*: as a result of expansion, the number of managers grows far beyond the

number that can efficiently report to a single person. At this stage, the organization must slow down its growth, and introduce a new formal and decentralized structure, organized by product/service line or by geographical area.

These three structures are uniform, in the sense that “all subordinates of the chief executive are structured either in an entrepreneurial, or a functional, or a product line or area structure” [71, p.188]. Beyond a certain size, this uniformity cannot be maintained: the decentralized structure need change into a *multistrukture*, i.e. a federated managerial structure where “different building blocks may be combined into different kinds of structures” [71, p. 189]. The Japanese, according to Weinshall, were the first to operate their large organizations in multistruktures, in a type of organization known as *zaibatsu*. The multistrukture is inherently flexible, in that it enables changes in the composition of the federated basic structures. This natural evolution as enterprises grow from the entrepreneurial structure to the multistrukture is depicted in Fig. 3. The actual values of the time on the abscissa and the logarithm of the size of the organization on the ordinate depend on the sector of activity of the company, and change dramatically from one type of industry to another.



**Fig. 4.** The five phases of organization development (adapted from Greiner [28, p. 39])

To conclude with enterprise management, let us take a look at the *evolutions and revolutions cycle* depicted in Fig. 4, an evolution trend that Greiner identified way back in 1972. With hindsight, it is amazing to see how this cycle, devised for enterprise management, suits network and systems management well. If the first three phases look similar to those identified by Weinshall, the last two, coordination and collaboration, are incredibly visionary, and predicted 25 years ago what intelligent agents are now striving to achieve in network and systems management! It is also interesting to notice that the last crisis is left as unknown. Would the combination of hierarchical and cooperative paradigms be the ultimate solution? Or was the idea of collaboration so new in the enterprise world that Greiner, lacking hard evidence, did not want to speculate on what could go wrong then?

#### 4.2. What Do We Learn From Enterprise Management?

What do Mullins, Weinshall and Greiner tell us that could apply to network and systems management? First, all the management paradigms they consider are hierarchical, except for the last two phases described by Greiner, characterized by a mix of hierarchical and cooperative paradigms. Likewise, most distributed network and systems management paradigms are hierarchical today, and cooperative paradigms have only just started to appear in hybrid structures similar to Weinshall’s multistruktures. Second, delegation schemes should evolve as enterprises grow in size, otherwise they become inefficient. Similarly, distributed network and systems management should rely on different distributed network and systems management paradigms as networks grow in size and complexity. In this respect, the recent explosion of new distributed paradigms seems justified, since networks have grown by at least an order of magnitude in terms of size and complexity since the SNMPv1 and OSI management frameworks were devised.

Third, if there are many ways of dividing up enterprise organization structures, depending on the granularity of the analysis, most authors agree with Weinshall that they all coalesce in three broad types: by function, by product or service, and by geographical area. We can see some similarities here between enterprise management and distributed network and systems management: the division of management domains by geographical area, for example, makes sense in both

worlds. But there are clear discrepancies too, since the basic entities are human beings in one case, and machines or programs in the other. The division by function only makes sense in the enterprise world: it takes long for a person to become an expert in accounting or electrical engineering, and an accountant cannot be turned into an engineer overnight; conversely, a computer can be equipped with new competencies in a matter of minutes or hours, by simply transferring a few programs and testing them on their new host. In section 4.2.1, we show that Mullins' federal decentralization and Weinshall's decentralized structure map onto our *delegation by domain* scheme in distributed network and systems management, whereas Mullins' functional decentralization and Weinshall's functional structure map onto our *delegation by task* scheme.

The fourth and most important thing we can learn from enterprise management is this common evolution trend, of which Greiner and Weinshall give two different, but compatible, flavors. There is a natural evolution of companies from centralized structures to decentralized ones, and from lightly decentralized structures (organized by function) to more decentralized ones, with independent units (organized by product/service or by geographical area), to even more decentralized ones (based either on federation or on cooperation). In network and systems management terms, these four stages nicely map onto the different types presented in our simple typology. The evolution which occurred in enterprise management over the 20th century suggests that the same evolution may take place in network and systems management in the next decade or so: the time scale may be different, but the evolution trend toward more distributed and cooperative management is the same.

#### 4.2.1. Delegation by Domain, Delegation by Task

How do the eight types identified by Mullins translate in network and systems management terms? We just saw that delegation by geographical domain applies equally well to both worlds, but what about the other types? In 1994, Boutaba<sup>1</sup> [8] identified a number of criteria to define domains in network and systems management. Resources are grouped into domains when they share a common feature, which may be the organizational structure (same department, same team), the geographical location, access permissions (resources accessible to a user, a group of users, or everybody), the type of resource (same vendor, same management protocol), the functionality of the resource (printer, mail system), etc. Some items in this list resemble Mullins' types, although they were made in very different contexts. But both of these lists are far too detailed for our typology. In network and systems management, we propose to group all possible delegation policies in just two types: delegation by domain and delegation by task.

*Delegation by domain* relies on static tasks: the manager at level (N) assumes that the manager at level (N+1) knows all of the management tasks to be completed within its domain. In today's networks, delegation by domain typically translates into delegation by geographical domain, to manage geographically dispersed enterprises. For instance, let us suppose that the headquarters of a multinational company are located in Sydney, Australia. This company cannot afford to manage its large subsidiaries in the USA, Asia or Europe over expensive and relatively slow transcontinental wide-area network links. Let us consider its European subsidiary, located in Geneva, Switzerland. The NMS in Sydney delegates the whole management of the Swiss subsidiary to the NMS located in Geneva, and expects it not to report a local printer going down, but to report that the number of errors per minute has exceeded a critical threshold on the Switzerland-Australia link. The point here is that the Australian NMS does not tell the Swiss NMS what to report: it expects it to be able to work it out by itself. In practice, this translates into a human being, the local network administrator, hard-coding in the Swiss NMS what to report back to Sydney, and how to manage the rest of the local network. There is no mechanism for the Australian NMS to alter the way the Swiss NMS manages its domain: it is a white-card type of delegation, where the Geneva-based NMS has total control over its own local network. Network management is not automated, and there is no way for the Australian network administrator to enforce a management policy over all its subsidiaries. Clearly, these are serious limitations.

*Delegation by task*, conversely, offers a finer grained vision at level (N) of the management processing occurring at level (N+1). A task is a network management application unit. It can be viewed at different scales: we show in section 4.2.2 that it makes sense to distinguish micro-tasks from macro-tasks. Before we go further on this, it is important to realize that an important property derives from the fact that the manager at level (N) can see the different tasks at level (N+1), as well as other tasks of its peers at level (N): tasks no longer need be static, hard-coded in every NMS. They may as well be dynamic, and modified on the fly. This idea was first applied to network management when the MbD framework was devised: Goldszmidt departed from the well-established notion of static tasks underlying the centralized paradigm, and introduced the notion of dynamic tasks, transferable from the NMS to its subordinate agents. This paradigm was soon generalized by others to transfer dynamic tasks from a manager at level (N) to a manager at level (N+1).

---

1. This study is largely based on earlier work by Sloman, Erstin, Gomberg, Nettet and the ANSA project (see references in [8, p. 89]).

#### 4.2.2. Micro-tasks and Macro-tasks

A manager at level (N) has several ways of driving a subordinate at level (N+1). With traditional approaches such as SNMPv1, the basic unit in the manager-agent dialog is the protocol primitive: the manager issues a series of Get and Set requests to the agent. The data manipulated are MIB variables, which are statically defined when the MIB is designed. With large MIBs or large networks, this leads to the micro-management syndrome [26], which entails a significant network overhead and a poor usage of resources (in the manager, in the agent, but also in the network equipment in between).

Recent approaches, conversely, avoid this syndrome by splitting the whole management application into many different units, or *tasks*, and by distributing these tasks over a large number of NMSs and agents, while still letting the manager at level (N) in control of what subordinates at level (N+1) do. The underlying mechanism of this distribution is independent of the tasks being delegated: it can rely on program transfer, message passing, RPCs, etc. What is relevant for the management application is the granularity of the delegation, that is, the way the work is divided. Clearly, there is a wide spectrum of task complexities, ranging from the mere addition of two MIB variables to the whole management of an ATM switch. We propose to distinguish only two levels in our enhanced typology: micro-tasks and macro-tasks.

A *micro-task* ( $\mu$ -task) just performs preprocessing on static MIB variables, typically to make statistics. It is the simplest way of managing site-specific, customized variables. There is no value in these data *per se*, they still need be aggregated by the NMS one level up. If contact with the NMS is lost, statistics are still gathered, but there is no way for the subordinate to take corrective action on its own. In the case of a *macro-task* (M-task), the entire control over an entity is delegated. A macro-task can automatically reset a network device, or build an entire daily report, etc. If contact is lost with the NMS one level up, corrective actions may be automatically undertaken.

For completeness, delegation by domain with dynamic tasks is considered in this article as a particular case of delegation by M-task: the domain then describes the scope of the task, and the tasks are explicitly defined (as opposed to the delegation by geographical area, where tasks are implicitly defined).

### 5. CRITERIA FOR AN ENHANCED TYPOLOGY

In [38], we studied the features that designers of network and systems management applications expect from strongly distributed management paradigms. We identified a number of selection criteria, and showed that besides interoperability and scalability, which are already addressed by weakly distributed management paradigms, the two most critical criteria for designers are the semantic richness of the information model, and the degree of automation of management allowed by a paradigm.

#### 5.1. Semantic Richness of the Information Model

The semantic richness of the information model of a management application is an indication of the expressive power of the abstractions used in this model. It measures how easy it is for designers of network and systems management applications to specify a task to be executed by an NMS or an agent. The higher the level of abstraction used to model a management application, the higher the semantic richness of the information model, and the easier it is for a human to build and design a management application.

Humans like to think at a high level of abstraction. But management frameworks have traditionally offered fairly poor Application Programming Interfaces (APIs), constraining designers to model management applications with low-level abstractions. This limitation has been addressed recently by some of the new management paradigms, as we show in this section. Today, designers of management applications have the choice between three types of abstractions to build their information model:

- managed objects, offering low-level abstractions
- computational objects, offering high-level abstractions
- goals, offering very high-level abstractions.

We are now going to present these three types of abstractions. We will introduce and compare the concepts of protocol API and programmatic API, and will identify a new criterion for our enhanced typology: the degree of specification of a task.



### 5.1.1. Managed Objects (Low-level Abstractions)

Both the SNMP and the OSI management frameworks offer a *protocol API*: in these frameworks, there is a one-to-one mapping between the communication model and the information model, to use the ISO/ITU-T terminology [13]. In other words, the abstractions defined in the information model, which constitute the building bricks for the designer of a network and systems management application, are identical to the protocol primitives used underneath. The communication protocol is not transparent to the management application: this breaks a well-established rule in software engineering. For instance, in the different SNMP frameworks, network programmers have to think in terms of SNMP GETs and SNMP SETs when they write a management application (typically with Perl [70] scripts).

We call this the *managed object* approach, since both the IETF and the ISO use this phrase to describe a basic unit of the information model in the SNMP and OSI management frameworks. This identity between the communication model and the information model has nothing to do with the protocol themselves: it is imposed by the management frameworks. The limitations entailed by this approach reflect in the apparent limitations of some paradigms, such as mobile code: today, in the Internet world, most mobile code technologies still use a derivative of the basic SNMPv1 API developed at the Carnegie Mellon University, PA, USA, in the early 1990s, which provides for the `snmpget` and `snmpset` commands. But nothing inherent in the mobile code paradigm itself prevents such technologies from using higher-level abstractions.

When a management application is designed with managed objects, a protocol is automatically imposed, the managed objects must live in fully-blown agents (in the case of TMN, these agents need implement a good deal of the OSI stack, including CMIP and CMIS), and the manager-agent style of communication is imposed. These are very strong constraints imposed on management application designers.

### 5.1.2. Computational Objects (High-level Abstractions)

Protocol APIs are based on ideas which started to be questioned in the 1970s. Since the mid 1980s, the software engineering community has been advocating the use of *programmatic APIs* instead, which have been one of the selling points of the object-oriented paradigm. With such APIs, any object belonging to a distributed system is defined by the interface it offers to other objects. The distributed object model is independent of the transport protocol: it only defines a programmatic interface between an invoker and operations (methods) supported by an object. This programmatic API relies on a communication protocol at an engineering level, but this protocol is completely transparent to the management application designer.

We call this the *computational object* approach, with reference to the terminology used in ODP and ODMA. In this approach, designers of management applications can rely on rich class libraries, offering high-level views of a network devices and systems. Few constraints are imposed on the design: objects may be distributed anywhere, they need not live in specific agents implementing specific protocol stacks. The only mandatory stack is the one implementing the distributed processing environment. No specific organizational model is imposed or assumed: the management application solely relies on object-to-object communication. The administrator may define his own site-specific classes, and use them in conjunction with libraries of classes implementing standard MIBs, such as Sun's transcription of MIB-II [45] in JMAPI.

### 5.1.3. Goals (Very High-level Abstractions)

The third type of abstractions that may be used in information models is the *goal*. In section 3.4, we saw that cooperative paradigms are goal-oriented: the management application is split into tasks, which are modeled with very-high level abstractions and partially specified with goals. Once these goals have been sent by the manager to the agent, it is up to the agent to work out how to achieve these goals. This approach is fundamentally different from the one taken by weakly or strongly distributed hierarchical paradigms, where the management application is broken down into fully specified tasks. Whether the implementation of the task relies on calls to protocol primitives or method calls on objects, the agent is given by the manager a step-by-step *modus operandi* to achieve its task.

Like computational objects, goals offer a programmatic API. But unlike computational objects, they do not require an object-oriented distributed system: they can also rely on an Agent Programming Language [58], such as KQML [21].

Goals represent the highest level of abstraction available to management application designers today. They rely on fairly complex technologies, based on intelligent agents, which are not always available on managed systems or network equipment; so there is still a market for simpler technologies that support computational objects, or even simpler

technologies that only support managed objects. But goals are a type of abstractions that make it possible to manage very complex networks, systems or services, for which simpler abstractions are not suited. They are particularly well suited to specify negotiation, load-balancing or resource usage optimization. We will come back to this in section 7.

As we saw, managed objects and computational objects rely on fully-specified tasks, whereas goals rely on partially-specified tasks. In other words, the semantic richness of the information model and the degree of specification of a task are tightly linked. We decided to retain the latter as a criterion for our enhanced typology, since it shows two very different ways of specifying tasks in a management application. But we must bear in mind that these two criteria are not independent.

management application unit (= information model abstraction)	managed object	computational object	goal
abstraction level	low	high	very high
where does it live?	MIB	object	intelligent agent
how do we access it from the management application code?	transport protocol primitives (SNMPv*, CMIP, HTTP)	method call	2 possibilities: 1) agent communication language primitive 2) method call
degree of specification	full	full	partial

Fig. 5. Semantic richness of the information model

## 5.2. Degree of Automation of Management

Up to a few years ago, the main motivation behind the automation of management was to relieve as much as possible network and systems support staff from the burden of constantly monitoring visually a Graphical User Interface (GUI) and fixing problems manually as they occur. As systems and networks grow in size and complexity by the year, administrators become more and more eager to automate their management: *ad hoc* manual management is not coping anymore. While he was advocating the use of MbD, Yemini was claiming that “management should pursue flexible decentralization of responsibilities to devices and maximal automation of management functions through application software” [75, p. 28].

Today, the need for more automation of management is also commanded by two factors: the deregulation of the telecommunications industry worldwide, and the explosion of new services offered to end-users, specially multimedia services. More and more actors are competing in the telecommunications market: monopolies (or near monopolies) gave way to a plethora of competing network operators, service providers, service traders, content providers, etc. So any service provision today is likely to cross several networks, managed by different companies, with equipment from several suppliers [1].

More and more services are being offered, too: mobile telephony, electronic commerce, video on demand, videoconference, teleteaching, telemedicine, etc. Videoconferences, for example, used to be booked by fax on an *ad hoc* basis. End users would contact support staff several days in advance; support staff would fax the single provider on the market (the local network operator); they would receive a reservation confirmation and an invoice within a day or so, sometimes less; and finally, they would inform the end-user that the booking has been made. This process was time-consuming, very inefficient, and allowed many sources of problems. Today, end-users want to deal directly with a service trader via a user-friendly GUI, get the best possible deal for a videoconference scheduled in a couple of hours, and make an electronic commerce transaction with a mouse click. Such demands are much more complex than they used to be, and require considerably more work than mere faxes. As the number of such transactions grows (from once a month to once an hour), and as the demands become more stringent (I do not want to book a videoconference for next week but for this afternoon), it happens more and more often that manual handling is simply not an option. Service management has to be automated (to offer the on-line GUI that the end-user expects), then network management has to be automated (e.g., to handle resource reservations and potential rerouting), and eventually systems management has to be automated (e.g., to provide for automatic failover for video-on-demand servers).

As we show in Fig. 6, micro-tasks poorly automate distributed network and systems management, but macro-tasks are very good at it, since they enable remote agents to take corrective actions independently from the NMS. Intelligent agents are typically used in negotiation, for example to get the best deal for a cross-Atlantic videoconference from competing



## 7. HINTS AND TIPS TO SELECT A MANAGEMENT PARADIGM OR TECHNOLOGY

In this section, we are going to show how designers of network and systems management applications may use our enhanced typology to select a paradigm, and possibly a technology. In particular, we will give several examples of situations where changes are required in the management application, due to new user needs or a growth in size of the enterprise.

### 7.1. What Does This Enhanced Typology Tell Us?

One thing we have shown throughout this article, and specially in section 5.2 (“Degree of Automation of Management”), is that there is no win-all solution. Depending on the size and complexity of the network, service or system to manage, some paradigms are better suited than others. Certain paradigms like mobile code encompass a very large number of technologies, and are in turn sub-classed into multiple paradigms: this yields a very wide variety of fine-grained design styles. Therefore, designers should not be constrained when they model management applications: among all available paradigms, they should select one which allows them to model the problem at hand in the most natural way. The days of protocol APIs, when the focus was on network management communications, are now over: management applications should rely on programmatic APIs instead, where the focus is on software development.

A second important thing is that among all the technologies we reviewed herein, some are supported commercially and have been well-tested and widely deployed, but others are still confined to the research community and unsupported, and yet others have hardly gone beyond the proof of concept. To manage production networks and systems, it is critical to rely on well-tested and well-supported technologies: designers should therefore carefully choose between different technologies those offering the best guarantees. If no such technology is available for the paradigm they selected, they should come back to vendors and tell them they are ready to pay for such technologies. This is no wishful thinking: in the telecommunications industry, more and more equipment vendors are supporting CORBA to manage their switches; this move was not driven by a new ITU-T standard, but by customer demand. Similarly, IETF resumed worked on SNMP with SNMPv3 because vendors were pushed by customers to support improved security and multi-tier hierarchical management, as SNMPv2 did not live up to expectations.

The third point which comes out of our enhanced typology is that several paradigms span over multiple quadrants, as Fig. 6 clearly shows. So different technologies claiming to support a given paradigm may actually offer fairly different degrees of automation, or a different semantic richness of the information model. Let us take an example. A good marketing campaign convinces a network administrator that mobile code is the right paradigm to manage his network. So, before delving into the design of a new, powerful management application, he decides to investigate the market in order to buy a mobile code technology. All vendors claim to sell the best product on earth, so what technology should he choose? With our typology, he can see at a glance that under the same name, mobile code, he can actually buy four very different types of technologies: some offering low-level semantics, others high-level semantics; some offering a high degree of automation of management, others a low level. This typology allows him to choose the technology offering the best value-for-money ratio, according to the relative weight he gives to the four selection criteria.

The fourth and last point is that we did not list all existing technologies in our enhanced typology: for strongly distributed management, only paradigms are depicted. The motivation for this choice is three-fold. First, we want to keep this typology readable. Second, technologies evolve so quickly and this market is currently so active that any such effort would be doomed to fail: such information would be obsolete as soon as it gets published (like price lists). Java, for instance, as blurred the boundaries between mobile code and distributed objects in the recent past, as we show in Fig. 6. Third, we believe that the criteria we selected and presented are reasonably easy to understand, and that potential buyers of such technologies should be able to decide where to locate a given release of a given technology in Fig. 6, based on a short technical description of it.

### 7.2. Examples

We are now going to present a series of examples showing how to use our typology in network management. These examples could be easily adapted to show how to use this typology in systems management. In all these examples, we will see that the relative weights given by the network and systems managers to the different selection criteria give a clear indication of the best suited paradigm(s).

In a small company, to manage a small local-area network or a small distributed system comprised of a dozen of machines, there is no need for an expensive technology offering a high level of automation with computational objects or even goals:

a cheap solution with managed objects and micro-tasks is enough. Even centralized management may be suited in this case.

Let us assume that this company develops, and opens small subsidiaries abroad. It is now a geographically-dispersed enterprise, but still has fairly simple needs (data network, no multimedia services). A weakly distributed hierarchical technology is well-suited: the required degree of automation is medium, and managed objects are sufficient to deal with simple needs. RMON is a good candidate. If the wide-area links to the remote subsidiaries are very expensive, MbD may be an even better solution.

As people in this company start using multimedia services on a more or less regular basis, there will come a point where manual handling is no longer an option: a higher degree of automation is required. Depending on the demands of the users, and the complexity of the services they use, cheap Java-based distributed object technologies may be sufficient. Otherwise, solutions based on intelligent agents may also be put in place, to cope with these new services on an *ad hoc* basis.

As this company develops and grows larger and larger, there comes a time when the number of entities to manage is so large that the management application starts getting too complex. It is getting hard to modify it, and any change may cause a new problem, due to unforeseen side-effects. The semantic richness of the information model is too poor: managed objects have become inadequate. Even for simple day-to-day management tasks, it is now time to use computational objects instead.

Complex tasks for complex services are still dealt with by intelligent agents. As new services start getting used, new intelligent agents are added on an *ad hoc* basis. As this company is now pretty large, intelligent agents are no longer restricted to dealing with complex services. They may be used for pattern learning, for example: they may dynamically learn what are the peak and slack hours of a Virtual Private Network (VPN) in an ATM network, and automatically readjust the bandwidth rented from the service provider so as to reduce the bill of the company.

Finally, if this company is later bought by a large multinational with tens of thousands of networked systems to manage, the degree of automation of management then becomes critical. Day-to-day network management should then entirely rely on distributed objects: managed objects should be banned. If day-to-day management was already based on distributed objects in the smaller company, the integration within a larger management application will be considerably easier.

In this large multinational, the number of requests for high-level services such as multimedia services will also soon increase, and the diversity of services used will also grow. This calls for fairly elaborate systems, based on a large number of intelligent agents, which nobody has already tested on a large scale so far, but that will probably be necessary in the not-so-distant future.

## 8. RELATED WORK

Although the literature offers many examples of typologies of organizational structures in other research fields such as enterprise management [2, 19, 28, 49, 71], oddly enough, fairly little has been published recently with respect to organizational structures in network or systems management.

Before the outbreak of strongly distributed management technologies, most authors [60, 63] just presented the different management frameworks adopted by the IETF with SNMPv\* and the ISO with OSI. Since then, several authors [46, 57] showed the advantages of one technology or one management framework over the centralized approach. Others studied a single family of paradigms [23]. But few authors, in fact, have already proposed full-scope typologies, covering the whole range of network and systems management paradigms.

Those we found were all based on a single criterion, the organizational model, like our simple typology. Hegering et al. [30, p. 121] quickly sketch a typology of integrated management paradigms, comprised of centralized, hierarchical and cooperative paradigms. But the meaning they give to cooperative management is very different from ours. According to them, “both approaches [OSI management and SNMPv2 framework] support a cooperative management of peer systems” [30, p. 198]; so they seem to consider cooperation as a manager-to-manager dialog only. Leinwand et al. [35] propose a typology of network management paradigms, more detailed than the previous one, but still fairly different from our simple typology: they focus on whether management databases are located in sub-level managers, or in the top-level manager only; and they ignore what we call strongly distributed hierarchical paradigms and cooperative paradigms.

In the literature, we found a single multi-criteria typology comparing technologies implementing different distributed paradigms, by Kahani and Beadle [33]. Unlike our enhanced typology, it does not attempt to cover the full scope of management paradigms. Instead, it only considers four distributed technologies, and compares them with the centralized approach. The authors selected seven criteria to perform their evaluation: architecture (what we call organizational model), communication method, polling method, polling interval, autonomy, extensibility and flexibility.

In summary, we believe that our simple typology is the first one to integrate the whole range of network and systems management paradigms proposed to date (full-scope typology), and that our enhanced typology is the first one to also propose multiple criteria to compare and weight the relative merits of all management paradigms and technologies (full-scope and multi-criteria typology).

## 9. CONCLUSION

In order to help designers of distributed network and systems management applications select the right paradigm and technology for a given network or a given distributed system, we proposed two typologies in this article, grouping all management technologies into a limited set of management paradigms.

First, we presented a *simple typology*, based on a single criterion: the underlying organizational model. This typology is comprised of four types: centralized paradigms, weakly distributed hierarchical paradigms, strongly distributed hierarchical paradigms and cooperative paradigms. We then reviewed the main technologies implementing each paradigm.

Second, we presented an *enhanced typology*, based on four criteria: the granularity at which the delegation process takes place (by domain, by micro-task or by macro-task); the semantics of the information model (managed object, computational object or goal); the degree of automation of management (high, medium, low); and the degree of specification of a task (full or partial). Finally, we showed how to use our enhanced typology to select a paradigm or a technology.

In the future, we intend to integrate technologies based on mobile code, distributed objects and intelligent agents, and to demonstrate that the coupling of hierarchical and cooperative paradigms can address network managers' perennial quest for ever richer semantics and ever more flexibility. In particular, we will try to apply this integrated model to the management of multimedia networks and services.

## ACKNOWLEDGMENTS

This research was partially funded by the Swiss National Science Foundation (FNRS) under grant 5003-045311. The authors wish to thank C. Gbaguidi, D. Hutchison, G. Pavlou, G.P. Picco and J. Schönwälder for their valuable suggestions and comments. Part of the material presented herein was published in the proceedings of the 8th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'97) [39].

## REFERENCES

1. S. Aidarous and T. Plevyak. "Principles of Network Management". In S. Aidarous and T. Pleviak (Eds.), *Telecommunications Network Management into the 21st Century*, pp. 1-18. IEEE Press, 1994.
2. M. Armstrong. *A Handbook of Personnel Management Practice*. 4th edition. Kogan Page, London, UK, 1991.
3. M. Baldi, S. Gai and G.P. Picco. "Exploiting Code Mobility in Decentralized and Flexible Network Management". In K. Rothermel and R. Popescu-Zeletin (Eds.), *Mobile Agents: Proc. 1st Int. Workshop (MA'97), Berlin, Germany, April 1997*. LNCS 1219:13-26, Springer-Verlag, Berlin, Germany, 1997.
4. M. Baldi and G.P. Picco. "Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications". To appear in Proc. 20th Int. Conf. on Software Engineering (ICSE'98), Kyoto, Japan, April 1998.
5. W. J. Barr, T. Boyd and Y. Inoue. "The TINA Initiative". In *IEEE Communications Magazine*, 31(3):70-76, 1993.
6. T. Berners-Lee, R. Fielding and H. Frystyk (Eds.). *RFC 1945. Hypertext Transfer Protocol -- HTTP/1.0*. IETF, May 1996.
7. BMC Software, Cisco, Compaq, Intel and Microsoft. *Industry Leaders Propose Web-Based Enterprise Management Standards Effort*. Press Release, July 1996. Available at <URL:http://wbem.freerange.com/wbem/pressrelease.htm>.
8. R. Boutaba. *Une architecture et une plate-forme distribuée orientée objet pour la gestion intégrée de réseaux et de systèmes* (in French). PhD thesis, Pierre & Marie Curie University, Paris, France, March 1994.
9. J. Case, M. Fedor, M. Schoffstall and J. Davin (Eds.). *RFC 1157. A Simple Network Management Protocol (SNMP)*. IETF, May 1990.
10. J. Case, K. McCloghrie, M. Rose and S. Waldbusser (Eds.). *RFC 1451. Manager-to-Manager Management Information Base*. IETF, April 1993.
11. J. Case, K. McCloghrie, M. Rose and S. Waldbusser (Eds.). *RFC 1901. Introduction to Community-based SNMPv2*. IETF, January 1996.
12. CCITT (now ITU-T). *Recommendation X.700. Data Communication Networks - Management Framework for Open Systems Interconnection (OSI) for CCITT Applications*. ITU, Geneva, Switzerland, September 1992.
13. CCITT (now ITU-T). *Recommendation X.701. Data Communication Networks - Information Technology - Open Systems Interconnection - Systems Management Overview*. ITU, Geneva, Switzerland, January 1992.

14. CCITT (now ITU-T). *Recommendation X.710. Data Communication Networks: Open Systems Interconnection (OSI); Management. Common Management Information Service Definition for CCITT Applications*. ITU, Geneva, Switzerland, March 1991.
15. CCITT (now ITU-T). *Recommendation X.711. Data Communication Networks - Open Systems Interconnection (OSI); Management. Common Management Information Protocol Specification for CCITT Applications*. ITU, Geneva, Switzerland, March 1991.
16. CCITT (now ITU-T). *Recommendation X.722. Data Communication Networks - Information Technology - Open Systems Interconnection - Structure of Information Management: Guidelines for the Definition of Managed Objects*. ITU, Geneva, Switzerland, January 1992.
17. DMTF. *Desktop Management Interface Specification*. Version 2.00. March 1996.
18. DMTF. *Common Information Model (CIM) Specification*. Version 1.1. September 1997.
19. D. Evans. *Supervisory Management: Principles and Practice*. 2nd edition. Cassell Educational Ltd, London, UK, 1986.
20. R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee (Eds.). *RFC 2068. Hypertext Transfer Protocol -- HTTP/1.1*. IETF, January 1997.
21. T. Finin, R. Fritzson, D. McKay and R. McEntire. "KQML as an Agent Communication Language". In N.R. Adam, B.K. Bhargava and Y. Yesha (Eds.), *Proc. 3rd Int. Conf. on Information and Knowledge Management (CIKM'94)*, Gaithersburg, Maryland, USA, November 1994, pp. 456-463. ACM Press, 1994.
22. S. Franklin and A. Graesser. "Is it an agent, or just a program?: a taxonomy for autonomous agents". In J.P. Müller, M.J. Wooldridge and N. R. Jennings (Eds.), *Intelligent Agents III, Proc. ECAI'96 Workshop (ATAL)*, Budapest, Hungary, August 1996. LNAI 1193:21-35, Springer-Verlag, Berlin, Germany, 1997.
23. A. Fuggetta, G.P. Picco and G. Vigna. "Understanding Code Mobility". Accepted for publication in *IEEE Trans. on Software Engineering*. Submitted July 1997.
24. J. Galvin and K. McCloghrie (Eds.). *RFC 1445. Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)*. IETF, April 1993.
25. G. Goldszmidt. *Distributed Management by Delegation*. PhD thesis, Columbia University, New York, NY, USA, December 1995.
26. G. Goldszmidt and Y. Yemini. "Distributed Management by Delegation". In *Proc. 15th Int. Conf. on Distributed Computing Systems (ICDCS'95)*, Vancouver, Canada, May 1995. IEEE Press, New York, NY, USA, 1995.
27. J. Gosling and H. McGilton. *The Java Language Environment: a White Paper*. Sun Microsystems, October 1995.
28. L.E. Greiner. "Evolution and Revolution as Organizations Grow". In *Harvard Business Review*, 50(4):37-46, 1972.
29. D. Harrington, R. Presuhn and B. Wijnen. *An Architecture for Describing SNMP Management Frameworks*. Internet draft, version 7 (work in progress). IETF, November 1997.
30. H.G. Hegering and S. Abeck. *Integrated Network and System Management*. Addison-Wesley, Wokingham, UK, 1994.
31. ISO WG4 N1851. *Open Distributed Management Architecture*. Working Draft 3. ISO, July 1995.
32. ITU-T. *Recommendation M.3010. Principles for a Telecommunications management network*. ITU, Geneva, Switzerland, May 1996.
33. M. Kahani and H.W.P. Beadle. "Decentralized Approaches for Network Management". In *ACM Computer Communication Review*, 27(3):36-47, 1997.
34. S. Krause and T. Magedanz. "Mobile Service Agents enabling 'Intelligence on Demand' in Telecommunications". In *Proc. IEEE Global Telecommunications conference (GLOBECOM'96)*, London, UK, November 1996. IEEE Press, New York, NY, USA, 1996.
35. A. Leinwand and K. Fang Conroy. *Network Management: a Practical Perspective*. 2nd edition. Addison-Wesley, Reading, MA, USA, 1996.
36. D.B. Levi and J. Schönwälder. *Script MIB. Definitions of Managed Objects for the Delegation of Management Scripts*. Internet draft, version 1 (work in progress). IETF, March 1997.
37. T. Magedanz and T. Eckardt. "Mobile Software Agents: a new Paradigm for Telecommunications Management". In *Proc. 1996 IEEE Network Operations and Management Symposium (NOMS'96)*, Kyoto, Japan, April 1996. 2:360-369. IEEE Press, New York, NY, USA, 1996.
38. J.P. Martin-Flatin and S. Znaty. "Annotated Typology of Distributed Network Management Paradigms". Technical Report SSC/1997/008, SSC, EPFL, Lausanne, Switzerland, March 1997.
39. J.P. Martin-Flatin and S. Znaty. "A Simple Typology of Distributed Network Management Paradigms". In A. Seneviratne, V. Varadarajan and P. Ray (Eds.), *Proc. 8th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'97)*, Sydney, Australia, October 1997, pp. 13-24.
40. S. Mazumdar. "Inter-Domain Management between CORBA and SNMP". In *Proc. 7th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'96)*, L'Aquila, Italy, October 1996.
41. S. Mazumdar and T. Roberts (Eds.). *Translation of GDMO Specification into CORBA-IDL*. Report of the XoJIDM task force, August 1995.
42. S. Mazumdar (Ed.). *Translation of SNMPv2 Specification into CORBA-IDL*. Report of the XoJIDM task force, September 1996.
43. K. McCloghrie. "The SNMP Framework". In *The Simple Times*, 4(1):9-10, 1996.
44. K. McCloghrie (Ed.). *RFC 1909. An Administrative Infrastructure for SNMPv2*. IETF, February 1996.
45. K. McCloghrie and M. Rose (Eds.). *RFC 1213. Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. IETF, March 1991.
46. K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt and Y. Yemini. "Decentralizing Control and Intelligence in Network Management". In A.S. Sethi, Y. Raynaud and F. Faure-Vincent (Eds.), *Integrated Network Management IV, Proc. 4th IFIP/IEEE Int. Symp. on Integrated Network Management (ISINM'95)*, Santa Barbara, CA, USA, May 1995, pp. 4-16. Chapman & Hall, London, UK, 1995.
47. M.A. Mountzia. "Intelligent Agents in Integrated Network and Systems Management". In *Proc. 1996 EUNICE Summer School on Telecommunications Services*, Lausanne, Switzerland, September 1996.
48. M.A. Mountzia and D. Bénech. "Communication Requirements and Technologies for Multi-Agent Management Systems". In A. Seneviratne, V. Varadarajan and P. Ray (Eds.), *Proc. 8th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'97)*, Sydney, Australia, October 1997, pp. 223-236.
49. L.J. Mullins. *Management and Organisational Behaviour*. 2nd edition. Pitman, London, UK, 1989.
50. *The New Encyclopaedia Britannica*. 15th edition. Micropaedia, 11:586(taxonomy). The New Encyclopaedia Britannica, Chicago, IL, USA, 1997.
51. *The New Encyclopaedia Britannica*. 15th edition. Micropaedia, 12:89-90(typology). The New Encyclopaedia Britannica, Chicago, IL, USA, 1997.
52. OMG. *The Common Object Request Broker: Architecture and Specification*. Revision 2.0. July 1995.
53. M. Post, C.C. Shen and J. Wei. "The Manager/Agency Paradigm for Distributed Network Management". In *Proc. 1996 IEEE Network Operations and Management Symposium (NOMS'96)*, Kyoto, Japan, April 1996. 1:44-53. IEEE Press, New York, NY, USA, 1996.
54. A.S. Rao and M.P. Georgeff. "Modeling rational agents within a BDI-architecture". In R. Fikes and E. Sandewall (Eds.), *Proc. Knowledge Representation and Reasoning (KR&R-91)*, San Mateo, CA, USA, April 1991, pp. 473-484. Morgan Kaufmann, 1991.

55. M.T. Rose. *The Simple Book: an Introduction to Internet Management*. 2nd edition. Prentice Hall, Englewood Cliffs, NJ, USA, 1994.
56. V. Sahin. "Telecommunications Management Network: Principles, Models and Applications". In S. Aidarous and T. Plevyak (Eds.). *Telecommunications Network Management into the 21st Century: Techniques, Standards, Technologies and Applications*. IEEE Press, New York, NY, USA, 1994.
57. J. Schönwälder. "Network Management by Delegation: from Research Prototypes towards Standards". In *Proc. 8th Joint European Networking Conference (JENC8)*, Edinburgh, Scotland, UK, May 1997.
58. Y. Shoham. "Agent-Oriented Programming". In *Artificial Intelligence*, 60(1):51-92, 1993.
59. J. Siegel. *CORBA Fundamentals and Programming*. Wiley, New York, USA, 1996.
60. M. Sloman (Ed.). *Network and Distributed Systems Management*. Addison-Wesley, Wokingham, UK, 1994.
61. J.M. Smith, D.J. Farber, C.A. Gunter, S.M. Nettles, D.C. Feldmeier and W.D. Sincoskie. *SwitchWare: Accelerating Network Evolution*. Technical Report MS-CIS-96-38, CIS Dept, University of Pennsylvania, USA, 1996.
62. F. Somers. "HYBRID: Unifying Centralised and Distributed Network Management using Intelligent Agents". In *Proc. 1996 IEEE Network Operations and Management Symposium (NOMS'96)*, Kyoto, Japan, April 1996. 1:34-43. IEEE Press, New York, NY, USA, 1996.
63. W. Stallings. *SNMP, SNMPv2 and CMIP: the Practical Guide to Network Management Standards*. Addison-Wesley, Reading, MA, USA, 1993.
64. P. Steenekamp and J. Roos. "Implementation of service management policies: Applying Intelligent Agent Technology". In *Proc. 1996 IEEE Network Operations and Management Symposium (NOMS'96)*, Kyoto, Japan, April 1996. 2:402-413. IEEE Press, New York, NY, USA, 1996.
65. Sunsoft. Java Management API Architecture. Revision A. September 1996.
66. D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall and G.J. Minden. "A Survey of Active Network Research". In *IEEE Communications Magazine*, 35(1):80-86, 1997.
67. D.L. Tennenhouse and D. Wetherall. "Towards an Active Network Architecture". *ACM Computer Communication Review*, 26(2):5-18, 1996.
68. S. Waldbusser (Ed.). *RFC 1271. Remote Network Monitoring Management Information Base*. IETF, November 1991.
69. S. Waldbusser (Ed.). *RFC 1757. Remote Network Monitoring Management Information Base*. IETF, February 1995.
70. L. Wall, T. Christiansen and R.L. Schwartz. *Programming Perl*. 2nd edition. O'Reilly & Associates, Sebastopol, CA, USA, 1996.
71. T.D. Weinshall and Y.A. Raveh. *Managing Growing Organizations: a New Approach*. Wiley, Chichester, UK, 1983.
72. C. Wellens and K. Auerbach. "Towards Useful Management". In *The Simple Times*, 4(3):1-6, 1996.
73. R. Wies, M.A. Mountzia and P. Steenekamp. "A Practical Approach Towards a Distributed and Flexible Realization of Policies Using Intelligent Agents". In A. Seneviratne, V. Varadarajan and P. Ray (Eds.), *Proc. 8th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'97)*, Sydney, Australia, October 1997, pp. 292-308.
74. M. Wooldridge and N.R. Jennings. "Agent Theories, Architectures and Languages: a Survey". In M. Wooldridge and N.R. Jennings (Eds.). *Intelligent Agents. Proc. ECAI-94, Workshop on Agent Theories, Architectures and Languages, Amsterdam, The Netherlands, August 1994*. LNAI 890:1-39. Springer-Verlag, Berlin, Germany, 1995.
75. Y. Yemini. "The OSI Network Management Model". In *IEEE Communications Magazine*, 31(5):20-29, 1993.
76. Y. Yemini, G. Goldszmidt and S. Yemini. "Network Management by Delegation". In I. Krishnan and W. Zimmer (Eds.), *Proc. IFIP 2nd Int. Symposium on Integrated Network Management (ISINM'91)*, Washington, D.C., USA, April 1991, pp. 95-107. North-Holland, Elsevier, Amsterdam, The Netherlands, 1991.
77. T. Yemini and S. da Silva. "Towards Programmable Networks". In *Proc. 7th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'96)*, L'Aquila, Italy, October 1996.
78. T. Zhang, S. Covaci and R. Popescu-Zeletin. "Intelligent Agents in Network and Service Management". In *Proc. IEEE Global Telecommunications Conference (GLOBECOM'96)*, London, UK, November 1996. IEEE Press, New York, NY, USA, 1996.

**Jean-Philippe Martin-Flatin** is currently preparing for a Ph.D. thesis at EPFL. From 1990 to 1996, he was with the European Centre for Medium-Range Weather Forecasts in Reading, England, where he worked in network and systems management, security, Web management and software engineering. From 1988 to 1990, he worked on the Geographic Information System of a large city in France. In 1986, he received an M.Sc. in EE and ME from ECAM, Lyon, France. His main research interest is in distributed network management. He is a member of the IEEE and the ACM.

**Simon Znaty** is a professor at ENST-Bretagne in Rennes, France, where he teaches and does research in telecommunication services engineering. From 1994 to 1996, he was a senior researcher with the Telecommunications Laboratory at EPFL in Lausanne, Switzerland. From 1993 to 1994, he worked on service creation and management with the Network Architecture Laboratory at NTT in Tokyo, Japan. In 1993, he obtained his Ph.D. degree in computer networks from ENST in Paris, France. He is a member of the IEEE, and is the author of three books on networking.

**Jean-Pierre Hubaux** joined EPFL in 1990, where he is now a full professor and co-director of the Institute for computer Communications and Applications, which employs 20 Ph.D. students. His areas of interest include service engineering and multimedia services. Prior to this, he spent 10 years in France with Alcatel, where he was involved in R&D activities, mostly in the area of switching systems architecture and software. He was also in charge of the introduction of artificial intelligence techniques, especially for system troubleshooting. He is a senior member of the IEEE and a member of the ACM.